Hardness and Approximation of the Survivable Multi-Level Fat Tree Problem

Hung Q. Ngo
Computer Science and Engineering
SUNY at Buffalo
Email: hungngo@cse.buffalo.edu

Thanh-Nhan Nguyen
Computer Science and Engineering
SUNY at Buffalo
Email: nguyen9@buffalo.edu

Dahai Xu
AT&T Labs – Research
Email: dahaixu@research.att.com

***Abstract*—With the explosive deployment of "triple play" (voice, video and data services) over the same access network, guaranteeing a certain-level of survivability for the access network is becoming critical for service providers. The problem of economically provisioning survivable access networks has given rise to a new class of network design problems, including the so-called** SURVIVABLE MULTI-LEVEL FAT TREE **problem** (SMFT)**.**

We show that two special cases of SMFT **are polynomial-time solvable, and present two approximation algorithms for the general case. The first is a combinatorial algorithm with approximation ratio** $\min\{\lceil L/2\rceil + 1, 2\log_2 n\}$ **where** L **is the longest Steiner path length between two terminals, and** n **is the number of nodes. The second is a primal-dual** $(2\Delta_s + 2)$**-approximation algorithm where** Δ_s **is the maximum Steiner degree of terminals in the access network. We then show that approximating** SMFT **to within a certain constant** $c > 1$ **is NP-hard, even when all edge-weights of** G **are** 1**,** $L \leq 10$**, and** $\Delta_s \leq 3$**. Finally, we experimentally show that the approximation algorithms perform extremely well on random instances of the problem.**

Keywords: Broadband access networks, Graph theory, Optimization, Survivable tree, Topology design.

I. INTRODUCTION

Internet service providers are actively deploying triple-play services, i.e., transmitting voice, video and data to end users over the same access network. The ongoing projects include U-Verse from AT&T, FiOS from Verizon, and all-in-together service packages from cable companies (like Comcast, Time Warner, Cablevision, etc.). With the increasing dependence on the access network, both service providers and end users are expecting higher networking reliability or survivability against transient failures of network elements. Access networks used to be the bottleneck of end-to-end survivability. For economic reasons, most access networks were designed with a tree-like topology rooted at a central office of the service provider. While a tree topology is fragile to edge failures, traditional dual-home design, i.e., connecting a downstream node to two upstream nodes for redundancy, could be prohibitively expensive. Alternatively, the appropriate addition of a limited number of redundant links to the tree can provide better survivability-cost tradeoff [1].

The problem of provisioning survivability to an access network is unique due to the special feature called "Fat tree." Usually, only the central offices can be equipped with the expensive routing capability [2], and the other terminals within the access network only have the basic switching capability (such as traffic aggregation using multiplexing and demultiplexing). Therefore, to recover from a failure of its upstream link, the terminal has to relay the traffic from another terminal of the same distance or lower distance to the root in the fat tree. A sample fat tree [3] is shown in Fig. 1.

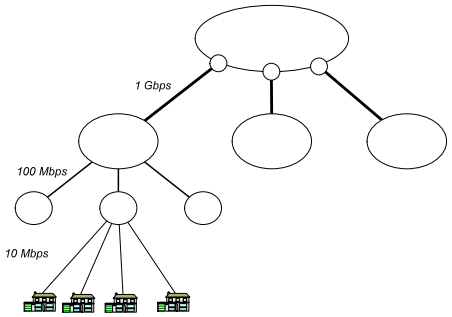

Fig. 1. Logical fat-tree architecture for access network

The SURVIVABLE MULTILEVEL FAT-TREE (SMFT) problem was defined in [1] as follows. We are given a graph $G = (V, E)$, called the *base graph*. Edges of G are weighted with non-negative weights $w : E \to \mathbb{R}^+$. The vertex set of G is partitioned into $T \cup S$, where vertices in T are called *terminal nodes*, and vertices in S are called *Steiner nodes*.

We are also given a (fat) tree $G_0 = (T, E_0)$ which is a subgraph of G that spans all terminal nodes and does not contain any Steiner node. There is one designated terminal node r called the *root*. For every terminal node $t \in T$, let P_t denote the path from t to r in G_0, and l_t – the *level* of t – be the length of P_t. (See Fig. 2.)

The objective is to augment G_0 with a minimum weight set of edges $A \subseteq E - E_0$ such that the augmented graph $G_0 \cup A = (V, E_0 \cup A)$ satisfies two constraints:

(i) *Bridge-connectivity constraint*: for every terminal $t \neq r$ there is a path P'_t in $G_0 \cup A$ from t to r which is edge-disjoint from P_t. We will refer to P'_t as a *backup path* for terminal t.

(ii) *Fat-tree constraint*: for each terminal $t \neq r$, P'_t does not contain any terminal v with $l_v > l_t$. (P'_t may or may not contain Steiner nodes.)

The *internal nodes* of a path are nodes in the path other than its two extreme ends. For any subgraph G' of G, an *arm*

of G' is a path in G' from a terminal t to another terminal u with no internal terminal. The arm is called a (t,u)-*arm*. If $l_t \geq l_u$ then the arm is also called a t-*arm*. Thus, a (t,u)-arm is either a t-arm, or a u-arm, or both (which can only happen when $l_t = l_u$). See Fig. 2 for an illustration.

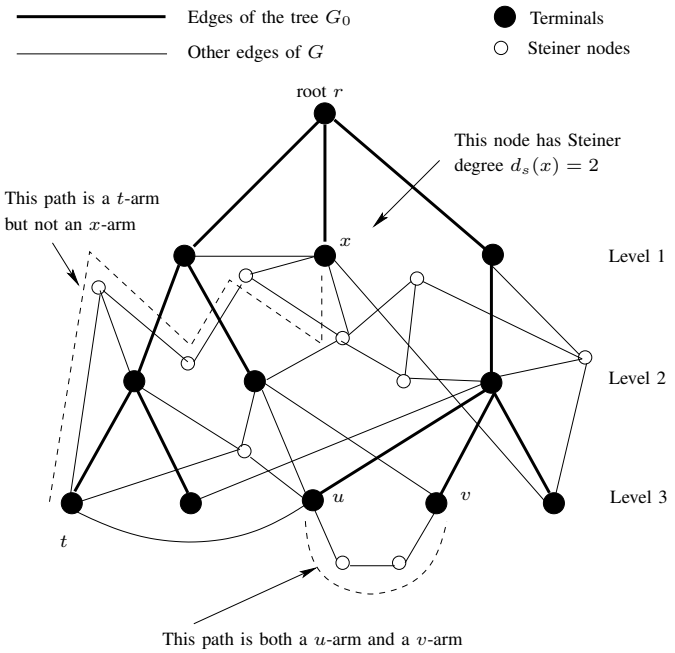

Fig. 2. Illustration of an SMFT instance and the *arm* concept.

Let L denote the longest arm length in G (in terms of number of edges, not in terms of edge weight). The *Steiner degree* $d_s(t)$ of a terminal t is the number of Steiner nodes adjacent to it. Let Δ_s denote the maximum Steiner degree of terminals in T.

The SMFT problem was shown to be **NP**-hard in [1]. Our contributions are as follows.

- We show that two special cases of SMFT are polynomial-time solvable. The first is when $G = K_n$ and edge weights are uniform. The second is when there's no Steiner node (i.e. $\Delta_s = 0$). The second algorithm serves as a subroutine for an approximation algorithm that follows.
- We present two approximation algorithms for SMFT. The first is a combinatorial algorithm with approximation ratio $\min\{\lceil L/2 \rceil + 1, 2\log_2 n\}$. This combinatorial algorithm also gives optimal solutions in the two cases above. Several natural variations of this combinatorial algorithms are also briefly discussed and analyzed. The second is a primal-dual algorithm with ratio $(2\Delta_s + 2)$.
- We show that approximating SMFT to within a certain constant $c > 1$ is **NP**-hard, even when all edge-weights of G are 1, $L \leq 10$, and $\Delta_s \leq 3$. This hardness result justifies the approximation ratios $\lceil L/2 \rceil + 1$ and $2\Delta_s + 2$ of the previous two algorithms, in the sense that for instances where there is some constant $c > 1$ such that a c-approximation does not exist (unless $\mathbf{P} = \mathbf{NP}$), our algorithms provide constant approximation ratios.
- We implemented the two approximation algorithms and a variation of the combinatorial algorithm and tested them on random instances of SMFT. The results are very encouraging as our algorithms give optimal or very-near-optimal solutions on these instances.

The rest of this paper is organized as follows. Section II presents algorithms for two special cases of SMFT where it is polynomial-time solvable. Section III gives two approximation algorithms, one purely combinatorial and the other primal-dual. Section IV proves a hardness of approximation result. Lastly, in Section V we show the performance evaluation of the approximation algorithms on random instances of the problem.

II. Two Polynomial-Time Solvable Special Cases

The following notations will be used throughout this section. Suppose the maximum level number is m. For each $i \in \{0, \ldots, m\}$, let T_i be the set of terminals at level i, and set $n_i = |T_i|$. For any subset F of edges, let $w(F)$ denote its total weight.

The following simple proposition gives a characterization of feasible solutions to SMFT, which provides an important insight into designing efficient (approximation) algorithms for this problem.

Proposition II.1. *An edge set $A \subseteq E - E_0$ is feasible if and only if, for every terminal $t \neq r$, there exists a t-arm Q_t in (V, A).*

Proof: Suppose A is feasible. For each terminal t, let P'_t denote a backup path for t. Let Q_t be the segment of P'_t from t to the first terminal v on P'_t. Due to the fat-tree constraint, $l_v \leq l_t$ as desired.

Conversely, suppose A satisfies the condition stated in the proposition. We prove by induction on the level l_t that each terminal t has a backup path. Consider a terminal t at level 1. If the t-arm Q_t ends at the root then we're done. Otherwise, Q_t ends at a terminal $v \neq t$ also at level 1. In this case, a backup path for t can follow Q_t to v and then take the edge (v, r) to the root. For the induction step, consider a terminal t at level $l_t > 1$. Suppose Q_t ends at terminal $v \neq t$. Consider two cases as follows. If v is an ancestor of t in the rooted fat tree, then a backup path for t is the concatenation of Q_t and P'_v, which exists by the induction hypothesis. When v is not an ancestor of t (but still $l_v \leq l_t$), let u be the least common ancestor of t and v in the tree G_0. A backup path for t can be constructed by following Q_t to v, then follow the $v \rightsquigarrow u$ path in G_0 to u, and finally take P'_u to the root. ■

A. Base Graph G is Complete with Uniform Edge Weights

In this subsection, we consider the case when $G = (V, E) = K_n$ and all G's edge weights are equal. Ignoring the trivial all-0 weight case, we can assume that $w(e) = 1$ for all e. The objective is to find a minimum number of edges to augment G_0.

Theorem II.2. *There is a linear-time algorithm which outputs optimal solutions to instances of* SMFT *where* $G = K_n$ *and* $w(e) = 1, \forall e \in E$.

Proof: Suppose $G = K_n$ and $w(e) = 1, \forall e \in E$. Let OPT denote the optimal augmentation cost for this instance. Consider three cases as follows.

Case 1: $\deg_{G_0}(r) \geq 2$. We first show that

$$\text{OPT} \geq \left\lceil \frac{n_1}{2} \right\rceil + \cdots + \left\lceil \frac{n_m}{2} \right\rceil. \tag{1}$$

Consider an optimal solution A^*. By Proposition II.1, there exists a t-arm Q_t in (V, A^*) for each terminal t at level $l_t \geq 1$. Let e_t be the edge incident to t on Q_t. Let $B = \{e_t \mid l_t \geq 1\}$, then OPT $\geq |B|$. If $e_u = e_v$, then u and v must be at the same level due to the fat-tree constraint. Hence, $|B| \geq \left\lceil \frac{n_1}{2} \right\rceil + \cdots + \left\lceil \frac{n_m}{2} \right\rceil$ as desired. We next show that the OPT's lower bound (1) can be achieved. Consider a level $i \geq 1$ in the tree G_0. Let $v_1^i, \ldots, v_{n_i}^i$ be the terminals at this level. Set $A = \emptyset$ initially.

- If $n_i = 1$, add the edge (v_1^i, r) to A. Since the root has degree ≥ 2, $n_i = 1$ implies $i \geq 2$. Thus, $(v_1^i, r) \notin G_0$.
- When $n_i \geq 2$, add the edge (v_{2j-1}^i, v_{2j}^i) to A for each $j = 1, \ldots, \lfloor n_i/2 \rfloor$. If n_i is odd, add the edge $(v_{n_i-1}^i, v_{n_i}^i)$ to A. We have added precisely $\lceil n_i/2 \rceil$ edges for level i. In light of Proposition II.1, it is not difficult to see that A is feasible with the desired cardinality.

Case 2: $\deg_{G_0}(r) = 1$ and there is at least one Steiner node. Similar to case 1, we can show that OPT $\geq 1 + \left\lceil \frac{n_1}{2} \right\rceil + \cdots + \left\lceil \frac{n_m}{2} \right\rceil$. The extra 1 is due to the fact that the node at level 1 needs an arm of length 2 to connect to the root. This lower bound can be achieved in much the same way as in case 1.

Case 3: $\deg_{G_0}(r) = 1$ and there's no Steiner node. In this case, the instance is infeasible because the terminal t at level 1 cannot have any t-arm. ∎

B. G_0 is a Spanning Tree of G (i.e. No Steiner Node)

In this subsection, we consider the case when the base graph G and its edge weights are arbitrary, and G_0 spans G.

Theorem II.3. *There is a polynomial-time algorithm which outputs optimal solutions to instances of* SMFT *for which* G_0 *spans* G.

Proof: Let E_i be the subset of edges of G which has one end-point at level i and the other at some level $\leq i$. Let $G_i = (V_i, E_i)$ be the subgraph of G induced by edges in E_i and their incident vertices. Note that $T_i \subseteq V_i$, and that the sets E_i are disjoint but the V_i may not.

We first observe the structure of an optimal solution. Let A^* be any optimal solution to an instance (G, G_0, w) of SMFT. Applying Proposition II.1, let Q_t be a t-arm in $G_0 \cup A^*$ from terminal t to another terminal v with $l_v \leq l_t$. Since there's no Steiner node, Q_t is precisely tv; namely, each Q_t is just an edge. For each $i \in [m]$, let $A_i^* = \{Q_t \mid l_t = i\}$. Then, for every i, A_i^* is a subset E_i such that every terminal in T_i is incident to some edge in A_i^*. In other words, T_i is "covered" by A_i^* (in the sense of edge-covering). Conversely, suppose for each i we pick a subset $A_i \subseteq E_i$ which covers T_i, then $A = \cup_{i=1}^m A_i$ is feasible by Proposition II.1. Since the E_i are disjoint, $w(A) = \sum_i w(A_i)$.

Consequently, to show that an optimal solution can be found in polynomial-time we only need to show, given a level i, how to choose a subset $A_i \subseteq E_i$ with minimum cost which covers T_i. This problem is similar to the *minimum weight edge-cover* (MWEC) problem on the graph G_i, which is polynomial time solvable (see, e.g., [4]). However, T_i is not necessarily the set of all vertices of G_i. Thus, our problem is slightly different because the standard MWEC problem requires covering all vertices of a graph.

Fortunately, we can easily transform our problem's instance to an instance of MWEC as follows. For each graph $G_i = (V_i, E_i)$, construct a graph $G_i' = (T_i, E_i')$ as follows. If an edge $e \in E_i$ has both ends in T_i, then add e to E_i' with the same weight. If an edge $e = (u, v) \in E_i$ has only one end $u \in T_i$, then add a loop (u, u) to E_i'. Set the loop's weight to be $w(e)$. Thus, all edges in E_i are "present" in E_i'. A subset of edges of E_i covers T_i iff it is an edge-cover of G_i' and vice versa. Thus, we only need to solve MWEC on G_i', for each i. (The standard reduction from MWEC to MINIMUM WEIGHT PERFECT MATCHING still works when the input instance of MWEC has loops.) ∎

III. Approximation Algorithms

A. A Combinatorial Approximation Algorithm

We will present several combinatorial approximation algorithms for SMFT. The first algorithm called CAA, shown in Fig. 3 uses as a subroutine the algorithm for the no-Steiner node special case of SMFT shown in Section II-B. The other combinatorial algorithms are variations of CAA which will be discussed later in the next subsection.

Before analyzing the algorithm, we need two structural lemmas. An optimal solution A^* is called *minimal* if there is no edge $e \in A^*$ such that $A^* - \{e\}$ is also feasible. (An optimal solution could be non-minimal when it contains some redundant 0-weight edges.)

Lemma III.1. *Let* A^* *be any minimal optimal solution to an instance* (G, G_0, w) *of* SMFT. *For each terminal* $t \in T - \{r\}$, *fix a* t*-arm* Q_t^* *in* (V, A^*). *Then, there exists a partition* $A^* = A_1^* \cup \cdots \cup A_k^*$ *of* A^* *satisfying the following properties:*

(i) *For each* $i \in [k]$, *the subgraph of* G *induced by* A_i^*, *denoted by* $G[A_i^*]$, *is a tree all of whose leaves are terminals and all of whose internal nodes (if any) are Steiner nodes.*

(ii) *For every terminal* $t \in T - \{r\}$, *the* t*-arm* Q_t^* *belongs to a unique* $G[A_i^*]$ *for some* $i \in [k]$

(It should be noted that a terminal t *may occur in several* $G[A_i^*]$.*)*

Proof: Consider the graph $G^* = (V, A^*)$. Remove from G^* all Steiner nodes that do not belong to any t-arm Q_t^*. Then, remove all terminals from G^*. The result is a collection of connected components $C_1, \ldots, C_{k_2}$. Each t-arm Q_t^* whose length is at least 2 must share a Steiner node with a unique

Algorithm CAA

Inputs: G, G_0 and edge weight function w

1: Construct another instance of SMFT on another graph $\bar{G} = (T, \bar{E})$ with edge weights $\bar{w} : \bar{E} \to \mathbb{R}^+$ as follows.
   - The fat tree G_0 for $\bar{G}$ is the G_0 for G
   - $\bar{G}$ will have no Steiner nodes
   - There is an edge uv in $\bar{E}$ if there is an arm connecting u and v in G
   - Set $\bar{w}(uv)$ to be the weight of the lightest arm (in terms of weight function w) connecting u and v in G
2: Run the algorithm in Theorem II.3 to get an optimal solution $\bar{A} \subseteq \bar{E} - E_0$ for the $\bar{G}$-instance
   // If the $\bar{G}$-instance is infeasible, then then G-instance is also infeasible.
3: $A \leftarrow \emptyset$
4: **for** each $uv \in \bar{A}$ **do**
5: Let Q_{uv} be the arm in G where $w(Q_{uv}) = \bar{w}(uv)$;
6: Add all edges in Q_{uv} to A
7: **end for**
8: **Return** A

Fig. 3. A Combinatorial Approximation Algorithm for SMFT.

component C_i, $i \in [k_2]$. For each $i \in [k_2]$, let A_i^* be the union of edges of the t-arms Q_t^* with lengths ≥ 2 which share a Steiner node with component C_i. Let k_1 be the number of t-arms Q_t^* with lengths 1. Put each of those k_1 t-arms in a separate set $A_{k_2+1}^*, \ldots, A_{k_2+k_1}^*$. Let $k = k_1 + k_2$. Then, $A_1^* \cup \cdots \cup A_k^*$ is a partition of A^* satisfying property (ii).

For property (i), we first show that each terminal is of degree 1 in $G[A_i^*]$. If there is a terminal t in $G[A_i^*]$ of degree ≥ 2, say tu and tv belongs to A_i^* where $u, v \in C_i$. Removing the edge tu does not affect the feasibility of A^*, because if tu is part of some arm then it can be replaced with tv and the $u \rightsquigarrow v$ path in C_i. This contradicts the minimality of A^*; thus, each terminal has degree 1 in $G[A_i^*]$. Next, suppose $G[A_i^*]$ is not a tree for some i. In this case, a cycle in $G[A_i^*]$ is a cycle in the component C_i. Again, removing an edge on this cycle does not affect A^*'s feasibility. Consequently, the partition satisfies property (i). ∎

Lemma III.2. *Let H be a rooted tree whose root is s and each of whose leaves v is associated with a positive integer l_v. Let D be the depth of H. Let v_0 be any leaf where l_{v_0} is smallest among all l_v. Let U be the set of all leaves v other than v_0 where $l_v = l_{v_0}$. There exists a collection $\mathcal{P}$ of paths in this tree satisfying the following properties:*

(i) *The path from v_0 to s is in $\mathcal{P}$*
(ii) *Each path in $\mathcal{P}$ other than the v_0, s-path is from a leaf to another leaf*
(iii) *For every leaf $u \neq v_0$, there is another leaf v with $l_v \leq l_u$ such that the u, v-path is in $\mathcal{P}$*
(iv) *If $U \neq \emptyset$, then there is a node $u \in U$ such that the u, v_0-path is in $\mathcal{P}$.*
(v) *Each edge of the tree H belongs to at most $D+1$ of the paths in $\mathcal{P}$.*

Proof: We induct on D. If $D = 1$, let $v_m, \ldots, v_0$ be the leaves of this tree. Without loss of generality, assume $l_{v_m} \geq \cdots \geq l_{v_0}$. The path collection $\mathcal{P}$ includes the v_m, v_{m-1}-path, ..., v_1, v_0-path, and v_0, s-path. (See Fig. 4.) It is easy to see that this path collection works.

Suppose $D \geq 2$. Let $s_m, \ldots, s_0$ be the neighbors of s and $H_m, \ldots, H_0$ be the subtrees rooted at $s_m, \ldots, s_0$, respectively. The depths of these subtrees are at most $D-1$. Without loss of generality, we can assume that v_0 belongs to H_0. For each $i \in [m]$, let v_0^i be a leaf in subtree H_i with lowest $l_{v_0^i}$ number among all leaves in H_i. Set $v_0^0 = v_0$. By rearranging the subtrees, we can assume that $l_{v_0^m} \geq l_{v_0^{m-1}} \geq \cdots \geq l_{v_0^0}$.

Apply the induction hypothesis to all the subtrees, and let $\mathcal{P}_i$ be the path collection for the subtree H_i. Let $\mathcal{P}$ be the union of the $\mathcal{P}_i$, ignoring the v_0^i, s_i-paths. Then, add into $\mathcal{P}$ the following paths: the v_0^m, v_0^{m-1}-path, the v_0^{m-1}, v_0^{m-2}-path, ..., the v_0^1, v_0^0-path, and the v_0, s-path. (See Fig. 4 for an illustration.) It is not difficult to verify that $\mathcal{P}$ satisfies all stated conditions. ∎

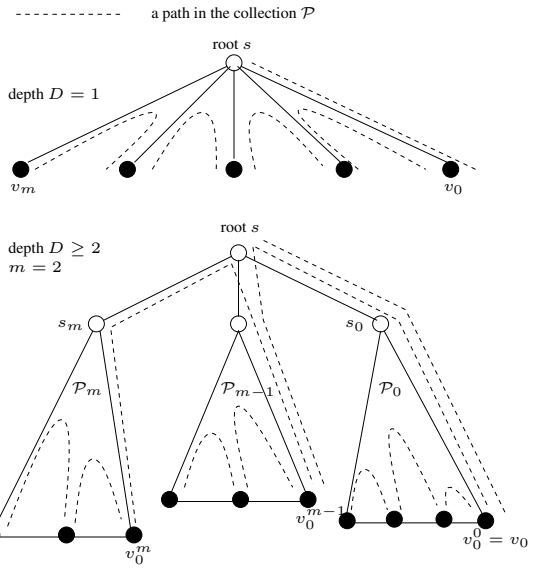

Fig. 4. Illustration of the proof of Lemma III.2

Theorem III.3. *Let L be the shortest length (in terms of the number of edges) of the arms in G, and n be the number of vertices of G. Then, Algorithm CAA shown in Fig. 3 has approximation ratio $\min\{\lceil L/2 \rceil + 1, 2\log_2 n\}$ and running time $O(n^2(m + n\log n))$, where $n = |V(G)|, m = |E(G)|$.*

Proof: Using the min-weight perfect matching algorithm from [5], the running time is certainly achievable. It might even be a little better if we use the more sophisticated algorithm from [6].

Let A^* be any minimal optimal solution to an instance (G, G_0, w) of SMFT, and let OPT $= w(A^*)$. Let A be the solution returned by Algorithm CAA. Let $\bar{A}$ be the solution returned in line 2 of Algorithm CAA. Note that $w(A) \leq w(\bar{A})$,

because by "unpacking" the edges in $\bar{A}$ into edges of G to put in the solution A, some edges of G might be duplicated.

Thus, it is sufficient to show two inequalities:

$$w(\bar{A}) \leq (2\log_2 n)\cdot \text{OPT} \quad (2)$$
$$w(\bar{A}) \leq (\lceil L/2\rceil + 1)\cdot \text{OPT} \quad (3)$$

The general strategy for showing $w(\bar{A}) \leq \alpha\cdot\text{OPT}$ is as follows. We show how to construct from A^* a feasible solution $\bar{B}$ to the $\bar{G}$-instance of SMFT where $\bar{w}(\bar{B}) \leq \alpha\cdot\text{OPT}$. Since $\bar{A}$ is optimal for the $\bar{G}$-instance, we conclude that $w(\bar{A}) \leq w(\bar{B}) \leq \alpha\cdot\text{OPT}$.

Let us first show inequality (2) using this strategy. For each terminal $t \in T - \{r\}$, fix a t-arm Q_t^* in (V, A^*). Consider the partition $A^* = A_1^* \cup \cdots \cup A_k^*$ satisfying the properties stated in Lemma III.1. As the A_i^* are disjoint, $\text{OPT} = \sum_{i=1}^k w(A_i^*)$.

Fix an $i \in [k]$. Let T_i be the set of terminals in $G[A_i^*]$. Color a terminal t in T_i *red* if its t-arm Q_t^* belongs to $G[A_i^*]$, and color t *white* otherwise. For each red terminal t in T_i, we will construct a t-arm Q_t^i in the tree $G[A_i^*]$ such that each edge in the tree belongs to at most $2\log_2 |T_i|$ of these Q_t^i.

To do so, let D_i be the directed graph obtained from the tree $G[A_i^*]$ by replacing each edge in $G[A_i^*]$ with two directed edges pointing to opposite directions. Then, D_i is an Eulerian graph. An Eulerian tour on D_i will visit each terminal in T_i exactly once and come back to the first terminal where the tour started. Let $t_1, \ldots, t_{|T_i|}$ be the sequence of terminals the tour visits, in that order. Following this tour, we obtain a collection of (undirected) arms in the tree $G[A_i^*]$: (t_1, t_2)-arm, (t_2, t_3)-arm, ..., $(t_{|T_i|}, t_1)$-arm, where each edge of the tree occurs in at most two arms.

Without loss of generality, assume t_1 has highest level among all terminals in this tree. For each $j = 1, 2, \ldots, \lfloor |T_i|/2 \rfloor$, a (t_{2j}, t_{2j+1})-arm is either a t_{2j}-arm or a t_{2j+1}-arm or both. In addition, the $(t_{|T_i|}, t_1)$-arm is always a t_1-arm. Thus, the above arm collection gives us t-arms for at least $\lceil |T_i|/2 \rceil$ of the terminals. The number of terminals t without a t-arm is at most $\lfloor |T_i|/2 \rfloor$. Next, we consider the terminals t without t-arms and follow the Euler tour again. The same argument gives us a collection of t-arms for at least half of these terminals t, where each edge of the tree occurs in at most two arms. In the end, we have constructed t-arms for all terminals, except for possibly the one at the lowest level, after at most $\log_2 |T_i|$ rounds. (Some t-arms for white terminals might have also been constructed, but we don't need them.) Each edge of the tree occurs in at most $2\log_2 |T_i|$ arms.

Note that, if there was a terminal whose level is strictly lower than all other terminals, then that terminal has to be white. Thus, if there is a lone remaining red terminal t in the end, then there must be at least another terminal t' at the same level as t. The t', t-path is a t-arm.

The above analysis gives us a collection of t-arms for every terminal t. Each t-arm is of the form $Q_t^{i_t}$ for a unique $i_t \in [k]$. Thus, the total weight of these t-arms is $\sum_t w(Q_t^{i_t}) \leq \sum_i 2\log_2 |T_i| w(A_i^*) \leq 2\log_2 n \sum_i w(A_i^*) = 2\log_2 n \cdot \text{OPT}$.

Now, consider a solution $\bar{B}$ for the $\bar{G}$-instance constructed as follows. For each t-arm $Q_t^{i_t}$, say $Q_t^{i_t}$ is a (t, u)-arm, inserts the edge tu into $\bar{B}$. By definition of the weight function $\bar{w}$, we have $\bar{w}(tu) \leq w(Q_t^i)$. The set $\bar{B}$ is certainly feasible for the $\bar{G}$-instance, and $\bar{w}(\bar{B}) \leq \sum_t w(Q_t^{i_t}) \leq 2\log_2 n \cdot \text{OPT}$.

We next prove inequality (3). For each red terminal in T_i, we will construct a t-arm Q_t^i in the tree $G[A_i^*]$ such that each edge in the tree belongs to at most $\lceil L/2 \rceil + 1$ of these Q_t^i. The construction strategy is different from the $2\log_2 n$-ratio case.

Consider a longest arm (in terms of the number of edges) in $G[A_i^*]$ from a terminal t_1 to a terminal t_2. Suppose this arm has length $L' \leq L$. (Recall that L denotes the length of a longest arm in the entire base graph G.) The case when $L' = 1$ is trivial, so we can assume $L' \geq 2$. Let s be a Steiner node in the middle of this longest arm. (If L' is odd, choose any one of the two middle nodes.) Fix s as the root of this tree. Then, the depth of this rooted tree is at most $\lceil L'/2 \rceil$.

By Lemma III.2, there exists a collection of paths satisfying the conditions stated in the lemma. If the set U in the Lemma is empty, then v_0 is the unique node with lowest level in the tree, and hence v_0 is white. The collection of paths given in the Lemma is the collection of t-arms we want. (The v_0, s-path is redundant, along with t-arms for white terminals.) If the set U is not empty, then the u, v_0-path in part *(iv)* of the lemma is a v_0-arm, so we are fine even when v_0 is red. ■

B. Three Variations of CAA

CAA-v1: In this variation, instead of computing the optimal solution to the $\bar{G}$-instance in line 2 of CAA, we just pick a shortest t-arm (in $\bar{G}$) for every terminal t and add to $\bar{A}$. The solution, if exists, is certainly feasible for the $\bar{G}$-instance. The analysis of CAA-v1 is almost identical to that of CAA, with a slightly worse approximation ratio of $\min\{\lceil L/2 \rceil + 2, 2\log_2 n + 1\}$. The advantage of CAA-v1 over CAA is that its running time is better. Essentially we only need to run an all-pair shortest-path algorithm such as the Floyd-Warshall algorithm in $O(n^3)$-time.

CAA-v2: In CAA we treat each level independently, and thus do not take into account the fact that a partial solution for a lower level might make the *incremental* cost for covering the next level smaller than the cost of covering the next level. This variation can be implemented by going through G_0 level-by-level, collapsing all terminals at lower levels and all used Steiner nodes, then find a cover for the current level. It is not difficult to see that the incremental cost for covering a level is at most the cost for covering it (without collapsing lower level nodes). Thus CAA-v2 has approximation ratio at least as good as CAA. The running time of CAA-v2 can be implemented so that it is the same as that of CAA-v1.

CAA-v3: Better yet, instead of computing the $\bar{G}$-edge cover for each level as in CAA-v2, we can make use of the optimal solution to the terminal cover problem in [1]. After collapsing lower levels and used Steiner nodes, we can treat it as a new root and the next level as terminals for the terminal backup problem. The approximation ratio is at least as good as CAA, but the worst-case running time is a lot worse: $O(\log(\text{OPT})n^{10})$.

Primal-Dual Algorithm for SMFT, Generic Version

$\mathbf{y} \leftarrow 0$ (all $y_{t,U}$ are set to 0)
$l \leftarrow 0$
$A_l \leftarrow \emptyset$
while A_l is not feasible **do**
  Choose a subset ν_l of terminal cuts which have not been "hit" by A_l
  Increase $y_{t,U}$ uniformly for all $C(t,U) \in \nu_l$ until there is some edge $e_l \in E' - A_l$ such that

$$\sum_{C(t,U)\in\nu_l:\ e_l\in C(t,U)} y_{t,U} = w(e_l)$$

  $A_{l+1} \leftarrow A_l \cup \{e_l\}$
  $l \leftarrow l+1$
end while
$A' \leftarrow A_{l-1}$ // start the *reverse delete* step
for $j \leftarrow l-1$ down to 1 **do**
  if $A' - \{e_j\}$ is feasible **then**
    $A' \leftarrow A' - \{e_j\}$
  end if
end for
Return A'

Fig. 5. General Description of a Primal-Dual Algorithm for SMFT. We will describe later how to choose (i.e. compute) ν_l efficiently.

C. A Primal-Dual $(2\Delta_s + 2)$-Approximation Algorithm

1) SMFT *under the Primal-Dual Framework:* Define $E' = E - E_0$ and $G' = (V, E')$. For any terminal $t \neq r$, and any subset $U \subset V$ such that U *does not* contain t but *does* contain all terminals with level $\leq l_t$, let $C(t,U)$ be the set of all edges $uv \in E'$ where $u \in U$, $v \notin U$, and each of u, v is either a Steiner node or a terminal at level $\leq l_t$. Intuitively, $C(t,U)$ is sort of like a cut $(U, \bar{U})$ of G' in which we ignore edges of the cut which are incident to some terminal with level $> l_t$. For lack of better words, we will call such an edge set $C(t,U)$ a *t-terminal cut*.

A simple (and standard) maxflow-mincut argument (see, e.g. [7]) along with Proposition II.1 leads to the following important proposition.

Proposition III.4. *An edge set $A \subseteq E'$ is feasible if and only if $|A \cap C(t,U)| \geq 1$ for every terminal $t \neq r$ and every t-terminal cut $C(t,U)$.*

The proposition allows us to view an instance of SMFT as an equivalent instance of the HITTING SET problem as follows. We need to choose a least-weight subset A of E' such that A "hits" every t-terminal cut $C(t,U)$ (i.e. $|A \cap C(t,U)| \geq 1$).

Associate a (dual) variable $y_{t,U}$ to each t-terminal cut $C(t,U)$. The generic primal-dual algorithm (Algorithm in Fig. 4.3 in [8], or Algorithm in Fig. 3 of [9]) can now be applied to SMFT, as shown in Fig. 5. In describing the algorithm, we have tried to use notations as consistently as possible with those used in [8], [9].

For any subset $A \subseteq E'$, a subset $D \subseteq E' - A$ is called a *minimal augmentation* of A if $A \cup D$ is feasible and for any $e \in D$, $A \cup D - \{e\}$ is not feasible. The following theorem gives the blueprint for deriving approximation ratio of the generic primal-dual algorithm.

Theorem III.5 (See [8], [9])**.** *If for every iteration l of the generic primal-dual algorithm,*

$$\max_{D:\text{min. aug. of } A_l} \sum_{C(t,U)\in\nu_l} |D \cap C(t,U)| \leq \alpha|\nu_l|, \qquad (4)$$

then the algorithm is an α-approximation algorithm.

2) Choosing the ν_l and Proving Approximation Ratio: Let A_l be the subset of edges at iteration l of the algorithm shown in Fig. 5. A terminal $t \neq r$ is said to be a *blocked terminal* (at iteration l) if there is no t-arm in (V, A_l). If A_l is infeasible then there must be at least one blocked terminal.

For each blocked terminal t, let H_t^l be the subgraph obtained from the graph (V, A_l) by removing all terminals at levels $> l_t$ (and their incident edges). Let $\bar{U}_t^l$ be the set of vertices reachable from t in H_t^l, and define $U_t^l = V - \bar{U}_t^l$. Since t is blocked, $\bar{U}_t^l$ contains t and possibly some Steiner nodes, but no other terminal. It is not difficult to see that $\bar{U}_{t_1} \cap \bar{U}_{t_2} = \emptyset$ for two different blocked terminals t_1 and t_2. Consider the graph $H_t^l[\bar{U}_t - \{t\}]$ which is the subgraph of H_t^l induced by $\bar{U}_t - \{t\}$. This graph has $c_l(t)$ connected components called the *private components* of t. It is easy to see that

$$c_l(t) \leq d_s(t) \leq \Delta_s.$$

Theorem III.6. *Suppose at each iteration l of the algorithm shown in Fig. 5 the set ν_l is chosen to be the collection of all terminal cuts $C(t, U_t^l)$ for which t is a blocked terminal. Then, the algorithm has approximation ratio $2\Delta_s + 2$ with running time $O(mn \log m)$, where $n = |V(G)|, m = |E(G)|$.*

Proof: The algorithm can be implemented using a union-find data structure in much the same way that Kruskal's MST algorithm is implemented, which justifies the running time.

We show that our choice of ν_l satisfies inequality (4) with $\alpha = 2\Delta_s + 2$. It is sufficient to show that, for any infeasible A_l and any minimal augmentation D of A_l,

$$\sum_{C(t,U_t^l)\in\nu_l} |D \cap C(t, U_t^l)| \leq (2\Delta_s + 2)|\nu_l|. \qquad (5)$$

To prove (5), we interpret the two sides of the inequality as combinatorial quantities. The right hand side of (5) is simply $(2\Delta_s+2)$ times the number of blocked terminals. The left hand side is slightly more complicated. Color edges in A_l *black* and edges in D *green*. Define $G_l = (V, A_l \cup D)$ and construct the graph $\bar{G}_l$ obtained from G_l by lumping together (i.e., identifying) all nodes in each set $\bar{U}_t^l$, retaining the possible parallel edges and loops introduced by this lumping process. Since the $\bar{U}_t^l$ are disjoint, the graph $\bar{G}_l$ is well-defined. For convenience, the node in $\bar{G}_l$ resulted from identifying all nodes in $\bar{U}_t^l$ is named $\bar{U}_t^l$, and is referred to as a *t-super node*. Thus, $\bar{G}_l$ has three kinds of nodes: t-super nodes for blocked terminals t, unblocked terminal nodes, and Steiner nodes. The

left hand side of (5) is simply the total green degree of the super nodes in $\bar{G}_l$.

To prove that the total green degree of super nodes is at most $(2\Delta_s + 2)$ times the number of blocked terminals, we devise a credit scheme. Each super node $\bar{U}_t^l$ is given $2c_l(t)+2$ "credits" to pay for the total green degree of the super nodes. Each green degree "debt" will be paid with one credit. Of the $2c_l(t) + 2$ credits of $\bar{U}_t^l$, there are two credits for each private component of t plus two extra credits. Let D_{in} denote the set of all green edges both of whose ends are inside some super node, and D_{ex} the rest of the green edges. Clearly edges in D_{in} do not contribute to the total green degree of super nodes. We will specify a way to write D_{ex} as a union

$$D_{\text{ex}} = \bigcup_{t \text{ is a blocked terminal}} D_t \tag{6}$$

and specify how the credits can be used to pay for the green degree debts contributed from each of the D_t.

Since $A_l \cup D$ is feasible, for every blocked terminal t there is at least one t-arm Q_t in G_l. The path Q_t induces a path $\bar{Q}_t$ in $\bar{G}_l$ from the super node $\bar{U}_t^l$ to either an unblocked terminal v or a super node $\bar{U}_v^l$, where $l_v \leq l_t$. We will also call $\bar{Q}_t$ a t-arm in $\bar{G}_l$. The t-arms in $\bar{G}_l$ may consist of some black edges and some green edges in D_{ex}.

Now, for each blocked terminal t, let $\bar{Q}_t$ be a t-arm in $\bar{G}_l$ with shortest length among all t-arms in $\bar{G}_l$; correspondingly, let Q_t denote a t-arm in G_l which induces $\bar{Q}_t$ in $\bar{G}_l$. The following is an important property of the $\bar{Q}_t$ chosen this way. **Claim:** All internal nodes of $\bar{Q}_t$ are either Steiner nodes of super nodes $\bar{U}_u^l$ with $l_u > l_t$.

Note the subtle point that this kind of internal super node is possible because the arm Q_t inducing $\bar{Q}_t$ can transit $\bar{U}_u^l$ via some private component of u but bypass u itself.

We now prove the claim. If there was a u-super node $\bar{U}_u^l$ with $l_u \leq l_t$ among the internal nodes of $\bar{Q}_t$, then the segment of $\bar{Q}_t$ from t to $\bar{U}_u^l$ is also a t-arm in $\bar{G}_l$ shorter than $\bar{Q}_t$, violating the definition of $\bar{Q}_t$. There cannot be any internal unblocked terminal u of $\bar{Q}_t$ because the arm would have ended at u. The claim is thus proved.

The union (6) is constructed algorithmically as follows. Initially, set $D_t = \emptyset$ for all blocked terminals t. Color all super nodes and private components in $\bar{G}_l$ *white*. Some private components may later be turned red, while others remain white. However, all super nodes will eventually be turned *red*. At each step, the algorithm does the following:

- add a few green edges from D_{ex} to some empty D_t,
- use the credits of some white super nodes and white private components to pay for the green degree "debts" imposed by the green edges just added to D_t,
- turn some super nodes' and private components' colors from white to red, including all the ones whose credits have been used up
- maintain the following **invariance**: for each blocked terminal t whose super node $\bar{U}_t^l$ is red, there exists a t-arm in $(V, A_l \cup D_{\text{in}} \cup (\cup_t D_t))$.

By the definition of blocked terminals, every terminal t that is **not** blocked has an entirely black t-arm in G_l. After the algorithm is terminated, all super nodes are red. The invariance thus ensures that $A_l \cup D_{\text{in}} \cup (\cup_t D_t)$ is feasible. Since D is a minimal augmentation of A_l, (6) must hold as promised. As all green degree debts have been paid by the credits given to the white nodes (at most $2\Delta_s + 2$ credits each), inequality (5) is thus proved.

It remains to describe the algorithm satisfying all of the above. The invariance certainly holds initially since there is no red super node yet. Process one by one the blocked terminals in *non-decreasing* order of their levels, breaking ties arbitrarily. Suppose a blocked terminal t is being considered.

If the super node $\bar{U}_t^l$ is already red, then do nothing and move on to the next blocked terminal. (In this case D_t remains the empty set.) The invariance still holds.

Now, suppose $\bar{U}_t^l$ is white. By the claim above, we only need to consider three cases as follows.

Case 1. $\bar{Q}_t$ has no internal super node. In this case, add all green edges in $\bar{Q}_t$ to D_t. The total green degree debt in D_t is at most 2 (could be 1 if $\bar{Q}_t$ ends at an unblocked terminal). We can thus use the two extra credits of $\bar{U}_t^l$ to pay for D_t, then color the super node $\bar{U}_t^l$ red. The invariance holds because Q_t is a t-arm in $(V, A_l \cup D_{\text{in}} \cup (\cup_t D_t))$.

Case 2. $\bar{Q}_t$ has some internal super node(s), and all of them are white. The private components of white super nodes are always white. In this case, add all green edges on $\bar{Q}_t$ to D_t. If $\bar{Q}_t$ has k internal super nodes, then the number of white super nodes on $\bar{Q}_t$ is at least $k+1$. (The other end of $\bar{Q}_t$ may be red.) The total green degree debt induced by D_t is at most $2k + 2$ (one on each end, and two for each internal node).

Now, Q_t will "grabs" on to at least one private component of each internal super node. We use the credits of these private components (2 each) to pay for the $2k$-debt amount and turn them red. Also color all internal super nodes red (their extra credits are not used). Then, use the two extra credits of $\bar{U}_t^l$ to pay for the green degree debts on two ends and turn $\bar{U}_t^l$ red also. Note that, by the claim above, all internal nodes $\bar{U}_u^l$ of $\bar{Q}_t$ satisfy $l_u > l_t$.

To show the invariance, note that Q_t itself is a t-arm in $(V, A_l \cup D_{\text{in}} \cup (\cup_t D_t))$. For the terminals u whose super nodes $\bar{U}_u^l$ have just been turned red, a u-arm can be constructed by going from u to a node in $\bar{U}_u^l$ that Q_t "touches," and then follow Q_t back to t itself. Since $l_u > l_t$, this is a legitimate u-arm.

Case 3. $\bar{Q}_t$ has some internal super node(s), at least one of which is red. As we move along Q_t from t to the other end, we either meet some red private component of an internal super node of we don't.

In the latter case, add all green edges on $\bar{Q}_t$ to D_t. We can then use the extra credits of $\bar{U}_t^l$ and the credits of white private components we see along the way to pay for the new debts, in the same fashion as the previous case. Turn $\bar{U}_t^l$ and the white private components red.

In the former case, let $\bar{U}_u^l$ be the first internal super node with a red private component that Q_t touches. Add all green

edges on the segment of $\bar{Q}_t$ from $\bar{U}_t^l$ to $\bar{U}_u^l$ to D_t. Turn all white nodes and private components on this segment to red. Similar to Case 2, the white nodes' and white private components' credits are sufficient to pay for the green degree debts incurred by D_t.

It remains to show that the invariance still holds. By the above claim, $l_u > l_t$; thus, u is not yet processed in our algorithm. The only way for a private component of $\bar{U}_u^l$ to turn red before u is processed is when that private component lies on some path Q_v where $\bar{U}_v^l$ is an already processed super node. Hence, $l_v \le l_t < l_u$. Thus, a t-arm can follow Q_t to the private component of u and then following Q_v back up to v. The handling of terminals whose super nodes just turned white is similar to that in Case 2. ■

IV. Hardness of Approximating SMFT

Theorem IV.1. *There exists a constant $c > 1$ such that it is* **NP***-hard to approximate* SMFT *to within a factor c, even when restricted to instances of* SMFT *for which $L \le 10$, $\Delta_s \le 3$, and all edge weights are* 1.

Proof: We present a *gap-preserving reduction* from MAX-E3SAT(5) to SMFT. (See, e.g., [10]–[12] for background on gap-preserving reductions and hardness of approximation.) An instance of MAX-E3SAT(5) consists of a CNF formula φ in which each clause C_i, $i \in [m]$, consists of exactly 3 literals and every variable belongs to exactly 5 clauses. Suppose there are n variables. It is easy to see that $m = 5n/3$. The objective is to find a truth assignment satisfying as many clauses as possible. Let $\text{OPT}(\varphi)$ denote the maximum number of clauses of φ satisfied by a truth assignment. It is known that it is **NP**-hard to distinguish between instances φ of MAX-E3SAT(5) for which $\text{OPT}(\varphi) = m$ and instances φ for which $\text{OPT}(\varphi) < (1-\epsilon)m$ for some fixed constant $0 < \epsilon < 1$ [13].

Our gap-preserving reduction from MAX-E3SAT(5) to SMFT is a polynomial-time algorithm satisfying the following properties:

(i) For each instance φ of MAX-E3SAT(5), the algorithm produces an instance $I = (G, G_0, w)$ of SMFT. This instance has $L \le 10$, $\Delta_s \le 3$, and $w(e) = 1$ for all $e \in E(G)$.

(ii) Let $\text{OPT}(I)$ denote the optimal cost of this instance. Then,

$$\begin{aligned} \text{OPT}(\varphi) = m \quad &\Rightarrow \quad \text{OPT}(I) \le 10n + m + 2 \\ \text{OPT}(\varphi) < (1-\epsilon)m \quad &\Rightarrow \quad \text{OPT}(I) > c(10n + m + 2) \end{aligned}$$

where $c = 1 + \frac{5\epsilon}{37} > 1$ is a constant.

Consider an instance φ of MAX-E3SAT(5) on variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$. Without loss of generality, we can assume that $n \ge 3$. The graph $G = (V, E)$ and a tree G_0 is constructed as follows. See Fig. 6 for an illustration.

The terminal set of G includes r, r', x_i for each $i \in [n]$, and C_j for each $j \in [m]$, where r is the root. The Steiner node set includes a node s, and $t_i^1, \ldots, t_i^9, f_i^1, \ldots, f_i^9$ for each variable x_i.

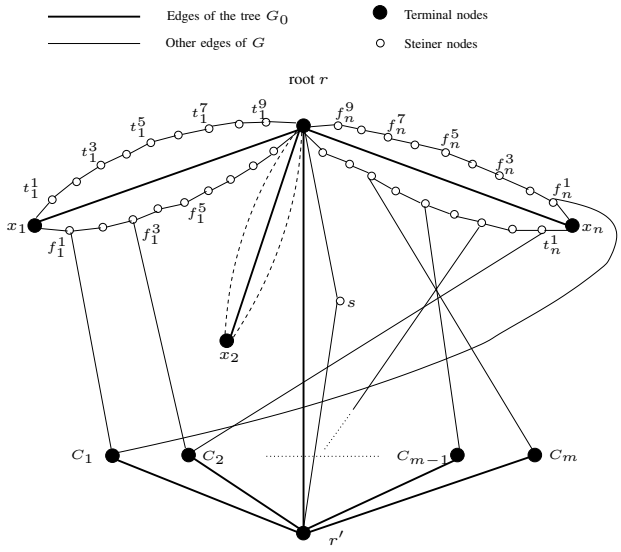

Fig. 6. Illustration of the gap-preserving reduction

The fat tree G_0 consists of the following edges: rx_i for each $i \in [n]$, rr', and $r'C_j$ for each $j \in [m]$. The rest of the edges of $E(G)$ involve Steiner nodes. First, there are two edges rs and sr'. Second, for each $i \in [n]$ there are two paths from x_i to r: the *true arm* $x_i t_i^1 \ldots t_i^9 r$ and the *false arm* $x_i f_i^1 \ldots f_i^9 r$. Third, there are three edges from each clause terminal C_j hooking C_j to either a true arm or a false arm corresponding to the literals that C_j contains. Specifically, consider a literal x_i. Let $C_{j_1}, \ldots, C_{j_k}$ ($k \le 5$) be the clauses that contains this literal, where $j_1 < j_2 < \cdots < j_k$. Connect C_{j_1} to t_i^1, C_{j_2} to t_i^3, ..., C_{j_k} to t_i^{2k-1}. (Since $k \le 5$, $2k-1 \le 9$.) For the literal $\bar{x}_i$, we connect the clauses that contains it to the false arm instead.

First, suppose $\text{OPT}(\varphi) = m$. Consider a truth assignment $a : \{x_1, \ldots, x_n\} \to \{\text{TRUE}, \text{FALSE}\}$ which satisfies all clauses of φ. Construct a feasible solution A to the SMFT instance as follows. For each variable x_i, if $a(x_i) = \text{TRUE}$ then add all edges on the true arm $x_i t_i^1 \ldots t_i^9 r$ to A; otherwise, add the false arm. For each clause C_j, add the edge from C_j to the arm of a literal which satisfies C_j. Also add two edges $r's$ and sr to serve as an r'-arm. It is easy to verify that A is feasible with cost $10n + m + 2$. The term $10n$ comes from one arm for each x_i. The term m comes from one edge for each C_j. And, the term 2 comes from edges $r's$ and sr. Thus, we have just shown that $\text{OPT}(\varphi) = m \Rightarrow \text{OPT}(I) \le 10n + m + 2$.

Second, suppose $\text{OPT}(\varphi) < (1-\epsilon)m$, namely no truth assignment can satisfy at least $(1-\epsilon)m$ clauses. Consider an optimal solution A^* to the SMFT instance. The r'-arm in (V, A^*) has to be $r'sr$ of weight 2. Since A^* is feasible, we can also fix an x_i-arm in (V, A^*) for each vertex x_i. Each of these arms is either a false arm or a true arm. Color all edges on these arms green. (It might be the case that both the true and false arms of some x_i belong to A^*. We just need to fix one green arm for such x_i.) Also fix a C_j-arm Q_{C_j} in (V, A^*) for each clause C_j. Let g be the number of Q_{C_j} which "grabs" on to one of the green arms. Then,

TABLE I
SUM COST OF THE EDGES CHOSEN BY VARIOUS ALGORITHMS FOR SMFT PROBLEM

Alg.	G_6^1	G_6^2	G_7^1	G_7^2	G_8^1	G_8^2	G_9^1	G_9^2	G_{10}^1	G_{10}^2	G_{11}^1	G_{11}^2
Opt.	145	101	81	93	147	145	117	90	108	103	74	72
CAA	145	103	81	93	148	161	117	90	112	103	80	75
CAA-v2	145	101	81	93	148	145	117	90	108	103	74	72
PD	145	101	81	95	148	148	117	90	109	104	74	72

it is not difficult to see that the total cost of A^* is at least $10n + 2 + g + 2(m - g) = 10n + 2 + 2m - g$.

Consider a truth assignment which sets x_i to TRUE if the true arm for x_i is green, and FALSE otherwise. This truth assignment satisfies at least g clauses. Thus, $g < (1 - \epsilon)m$. Consequently, noticing that $m = 5n/3$ and $n \geq 3$, we have

$$\begin{aligned} \text{OPT}(I) &= w(A^*) \geq 10n + 2 + 2m - g \\ &> 10n + 2 + 2m - (1 - \epsilon)m \\ &= \left(1 + \frac{\epsilon m}{10n + m + 2}\right)(10n + m + 2) \\ &= \left(1 + \frac{\epsilon 5n/3}{35n/3 + 2}\right)(10n + m + 2) \\ &\geq \left(1 + \frac{5\epsilon}{37}\right)(10n + m + 2). \end{aligned}$$

■

V. PERFORMANCE EVALUATION

We present the numerical results of solving the min-cost SMFT problem with various algorithms. We use BRITE [14] to create Waxman [15] topologies where the nodes are uniformly distributed on the plane and the probability of having an edge between any two nodes is inversely proportional to the exponential value of their Euclidean distance. We start with any node as the root and find the shortest path tree originated from it. Then we randomly choose another node, t, as well as all the nodes along the shortest path from t to root as terminals. We repeat the above procedure until we have collected enough terminals as required. The residual nodes are treated as Steiner nodes. The subgraph of the shortest path tree spanning all the terminals is considered as the existing multi-level access tree.

Starting with a 25-node Waxman network, we create 10 test instances with different number of Steiner nodes for different roots. Denote G_k^i as the i-th instance (different root) with k Steiner nodes.

Table I shows the total cost returned by various algorithms, which include optimal solution using Integer Linear Programming (ILP), CAA, CAA-v2, and the primal-dual algorithm. The ILP formulation is the ILP of the HITTING SET formulation of SMFT presented in Section III-C, which has exponential size in the input instance. We do not know of a better (i.e. polynomial-sized) ILP formulation for SMFT. This is the reason we couldn't test the approximation algorithms on larger random instances. From the table, we observe that all approximation algorithms can achieve near-optimal solution. In particular, CAA-v2 finds all the optimal solutions except for the instance of G_8^1.

In terms of running time, our approximation algorithms just take a few seconds. As expected, finding the optimal solution with ILP requires more time. For example, the lintprog in MATLAB quires 10-20 minutes for a 6-Steiner-node network, more than one hour for a 9-10 Steiner-node network, and more than two hours for an 11 Steiner-node network.

ACKNOWLEDGEMENTS

The work of Hung Q. Ngo was supported in part by NSF CAREER Award CCF-0347565.

REFERENCES

[1] D. Xu, E. Anshelevich, and M. Chiang, "On survivable access network design: Complexity and algorithms," in *Proceedings of the 27rd IEEE Conference on Computer Communications (INFOCOM)*. Phoenix, Arizona, U.S.A.: IEEE, Apr 2008, pp. 709–717.

[2] M. Andrews and L. Zhang, "The access network design problem," in *IEEE symposium on Foundations of Computer Science, FOCS'98, Palo Alto, CA*, 1998, pp. 40–59.

[3] A. Patzer, "Algorithms for a reliable access network," Master's thesis, Princeton University, 2004.

[4] A. Schrijver, *Combinatorial optimization. Polyhedra and efficiency. Vol. A*, ser. Algorithms and Combinatorics. Berlin: Springer-Verlag, 2003, vol. 24, paths, flows, matchings, Chapters 1–38.

[5] H. N. Gabow, "Data structures for weighted matching and nearest common ancestors with linking," in *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1990, pp. 434–443.

[6] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for general graph-matching problems," *J. Assoc. Comput. Mach.*, vol. 38, no. 4, pp. 815–853, 1991.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[8] M. X. Goemans and D. Williamson, "The primal-dual method for approximation algorithms and its application to network design problems," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed. PWS Publishing Company, 1997, pp. 144–191.

[9] D. P. Williamson, "The primal-dual method for approximation algorithms," *Math. Program.*, vol. 91, no. 3, Ser. B, pp. 447–478, 2002, iSMP 2000, Part 1 (Atlanta, GA).

[10] V. V. Vazirani, *Approximation algorithms*. Berlin: Springer-Verlag, 2001.

[11] L. Trevisan, "Inapproximability of combinatorial optimization problems," The Electronic Colloquium in Computational Complexity, Tech. Rep. 65, 2004.

[12] S. Arora and C. Lund, "Hardness of approximation," in *Approximation Algorithms for NP-Hard Problems*, D. Hochbaum, Ed. PWS Publishing Company, 1997, pp. 399–346.

[13] U. Feige, "A threshold of $\ln n$ for approximating set cover (preliminary version)," in *Proceedings of the Twenty-eighth Annual ACM Symposium on the Theory of Computing (Philadelphia, PA, 1996)*. New York: ACM, 1996, pp. 314–318.

[14] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Proc. of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems- MASCOTS'01*. Cincinnati, Ohio, August, Aug. 2001.

[15] B. M. Waxman, *Routing of multipoint connections*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1991.